Book Section:
Kennedy, J.V., Austin, J. orcid.org/0000-0001-5762-8614, Pack, R. et al. (1 more author) (1995) C-NNAP - A parallel processing architecture for binary neural networks. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN 95). (University of Western Australia, Perth, Australia, Nov 27-Dec 01, 1995). IEEE , New York , pp. 1037-1041.

https://doi.org/10.1109/ICNN.1995.487564

Published paper
Kennedy, J.V., Austin, J., Pack, R. and Cass, B. (1995) *C-NNAP - A parallel processing architecture for binary neural networks.* In: Proceedings of the IEEE International Conference on Neural Networks (ICNN 95). (University of Western Australia, Perth, Australia, Nov 27-Dec 01, 1995). IEEE, New York, pp. 1037-1041.

C-NNAP - A Parallel Processing Architecture for Binary Neural Networks

John V. Kennedy, Jim Austin, Rick Pack & Bruce Cass

Advanced Computer Architecture Group,
Department of Computer Science,
University of York, Heslington,
York, YO1 5DD, UK
johnk@cs.york.ac.uk

ABSTRACT

This paper describes the C-NNAP machine, a MIMD implementation of an array of ADAM binary neural networks, primarily designed for image processing. C-NNAP comprises an array of VME cards each containing a DSP, SCSI controller and a new design of the SAT peripheral processor. The SAT processor is a dedicated hardware implementation that performs binary neural network computations. The SAT processor yields a potential speed-up of between 108 times to 182 times that of the current DSP with its dedicated coprocessor. C-NNAP in association with the SAT provide a fast, parallel environment for performing binary neural network operations.

1. Introduction

Binary neural networks based on the N-tuple method [6] have been used in image processing for pig evisceration [1], scene analysis [5] and they also have potential for use in knowledge manipulation [3]. The binary neural network described in this paper is the Advanced Distributed Associative Memory (ADAM) developed by Austin [5]. An obstacle to the use of ADAM in real-time image processing systems is the requirement to process large quantities of data. To implement a *high performance* parallel version of ADAM we have developed the Cellular Neural Network Associative Processor, C-NNAP which is a MIMD machine with dedicated hardware assistance. The use of the C-NNAP machine for object recognition is explained in [2].

In the ADAM recall phase the majority of the computational effort is spent performing binary matrix multiplications that are implemented through binary summations. Although a coprocessor had previously been constructed to assist with the summing [10], the implementation has limited functionality. To improve the functionality the Sum And Threshold processor, SAT Version 1, was developed [8]. Analysis of this design highlighted a number of limitations that have been overcome in the design described in this paper, SAT Version 2. Because ADAM is a superset of the basic N-tuple method the SAT processor can also be used as a dedicated N-tuple pattern recognition processor.

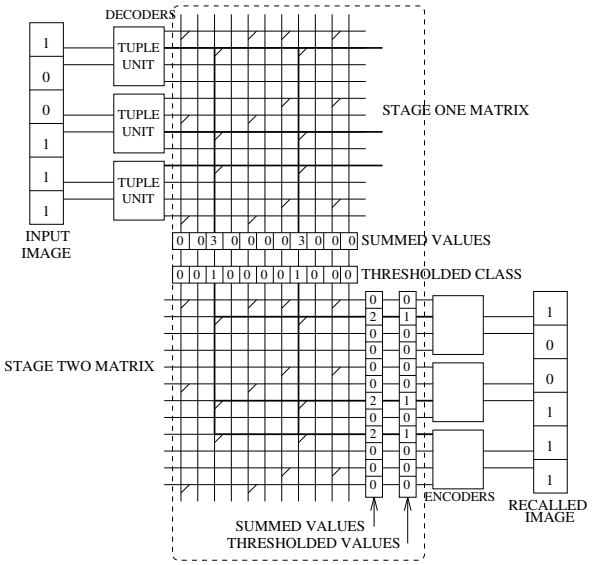
Section 2 of this paper explains the recall phase of the ADAM algorithm, Section 3 gives an overview of the C-NNAP machine and Section 4 is a detailed explanation and analysis of the SAT processor.

2. The ADAM Algorithm

The ADAM algorithm is a neural associative memory that has been used in a wide range of image analysis tasks [4]. The advantages of neural associative memories over traditional content addressable memories is that they can operate on noisy data. The ADAM algorithm is a significant improvement over other neural associative methods, ie. Willshaw [11], as it provides improved speed of operation, image storage ability and generalisation properties [5]. A complete description of the operation of the ADAM can be found in [5].

The SAT processor is primarily involved with the recall phase of the ADAM neural network as shown in Figure 1. The operation is initiated by applying the input image to the tuple units. The tuple units perform a n input to 2^n outputs logic decoder operation (n is 2 in the Figure) to activate a

Fig. 1: ADAM Recall



single output (shown with a bold line for the inputs shown). The active lines (the bold horizontal lines on the figure) cross set links (indicated by diagonal lines between the vertical and horizontal lines). The number of activated set links is summed at the base of the vertical lines to give the summed values. The summed values are thresholded using L-max thresholding, ie. the L highest summed values are set to 1, all the others are set to 0. This recalls the intermediate class pattern (L=2 in Figure 1). The class pattern is applied to the second stage matrix which activates links (shown with a bold line) which are summed in an analogous way to stage one. These summed values are thresholded using Willshaw thresholding, ie. those summed values which are equal to the number of class bits set (see Section 4) are set to 1, all others are set to 0. The output from the thresholding stage is passed to logic encoders whose output is the recalled image.

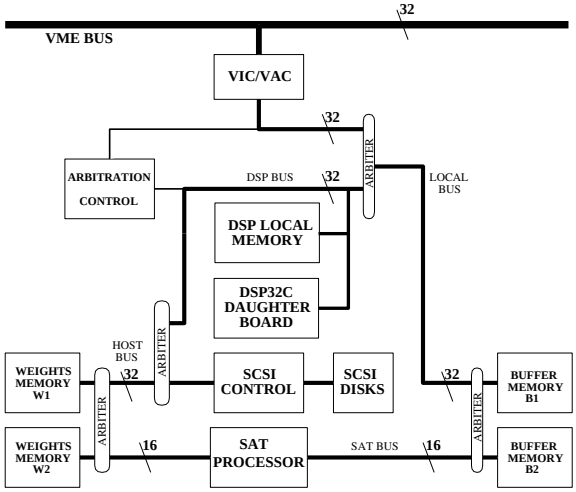
3. The Cellular Neural Network Associative Processor

C-NNAP is a VME based system with three basic building blocks: the S-Node, C-Node(s) and the I-node. The controlling workstation is the S Node (Supervisor node). The S node controls C node(s) (C-NNAP nodes) by providing them with address information and control programs. The I-node (Information node) is the I/O memory and data acquisition system from which the S-Node reads in the input data.

3.1. The C-NNAP Node

The architecture of a C node is shown in Figure 2 which has been designed to allow pipeline execution of the ADAM network. The daughter boards which hold the DSP and SAT processors have been designed to allow them to be upgraded independently of the C nodes. The C node has three, 25ns static RAM memories:

Fig. 2: The C Node Architecture



Weights Memory: The weights memory is divided into two independently accessible areas. The first area is used by the DSP to store the weights after their calculation whilst the second area contains the weights that the SAT processor will use during the recall operations. The switching between memories is controlled by the DSP.

Buffer Memory: The buffer memory is also divided into two independently accessible areas. The first area is used to store the non-tupled image prior to processing by the DSP and the tupled data prior to switching the memory into the SAT address area. This area can also be accessed by other nodes on the VME bus. The second area is used by the SAT as temporary storage and for storing results. The switching between memories is again controlled by the DSP.

DSP Memory: Only the DSP has access to the DSP memory which it uses as a program store and temporary storage.

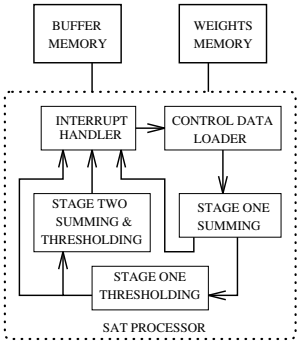
DSP Daughter Board: The DSP daughter board hosts an AT & T DSP32C clocked at 50 MHz. The DSP32C can be considered a slow processor, but the processing power required of the DSP is not extensive as the computationally intensive work is done by the SAT processor.

4. The Sum And Threshold Processor

This Section describes Version 2 of the SAT processor that performs all of the operations within the dotted box of Figure 1. The SAT processor is implemented using an Actel A1280XL FPGA and an Actel A1425A FPGA. The SAT is a peripheral processor that operates in parallel with the DSP thus releasing it to perform preprocessing and data movement operations.

This Version of the SAT processor is significantly different to the Version 1 design [8] as it overcomes the stage two summing bottleneck and it no longer uses a local memory. The SAT has access to the weights memory that contains the correlation matrices and the buffer memory that holds the control information, tuple pointers, summed values and recalled patterns. The block diagram of the SAT processor is shown in Figure 3.

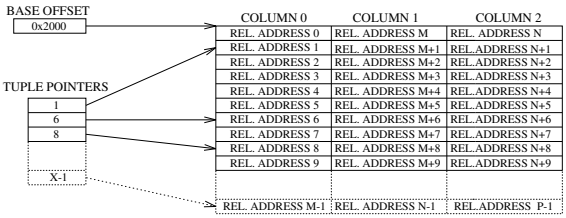
Fig. 3: SAT Block Diagram



Stage One Summing

The SAT has sixteen 16-bit registers (counters) for summing. The counters operate in parallel, thus allowing sixteen bits of the binary matrices to be summed in one clock cycle. The stage one binary matrix is stored as sixteen bit words, as shown in Figure 4 where each column is 16 bits wide, so that the data can be accessed in the format required by the 16 summing counters. The SAT uses tuple pointers generated by the DSP, from the input image, to calculate which lines of the stage one matrix require summing. To calculate the exact address of the weights an offset is provided in the control data. When the tuple pointer values are added to the offset the correct column and line of weights can be accessed. When the SAT sums the next column of the matrix it adds the length of the column in the matrix to the original offset, this results in the new offset for the next column. An example of this is shown in Figure 4, where the lines 1, 6, 8 etc. are the lines to be summed relative to the offset, eg. the first set of weights to be summed is at location 0x2001 (the offset + 1), the next weights are at 0x2006, etc. At the end of the column the length of the column is added to the offset, in this case M, to give the new start location. When the required weights in the weights memory have been accessed the summing counters are clocked. This has the effect of incrementing those counters whose input is an active weight. When all the active lines in a column have been summed the sixteen summed values are stored in the buffer memory.

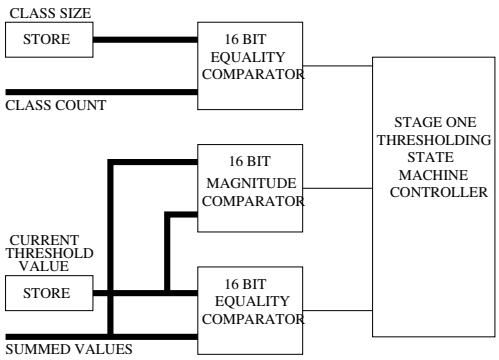
Fig. 4: Weights Address Calculation



Stage One Thresholding

When all the summed values have been written to the buffer memory, L-max thresholding (Section 2) is applied to them. The hardware used for this is shown in Figure 5.

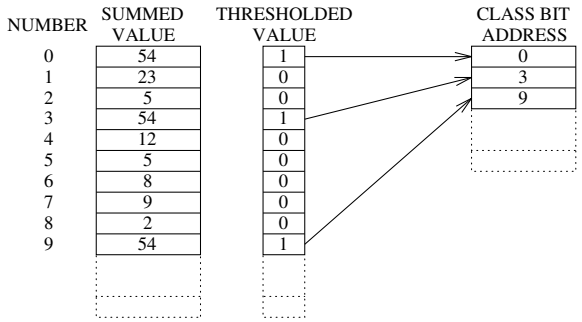
Fig. 5: Stage One Thresholding Block Diagram



Thresholding begins by inspecting all the summed values to find the maximum value stored, this becomes the current threshold value. All of the summed values are then checked, when a summed value equals the current threshold value this indicates that the corresponding class bit should be stored for use by the stage two summing controller. If insufficient class bits were found after the first thresholding iteration then the operation is repeated by finding the next highest summed value and using this as the new threshold value. This is repeated until all L class bits have been recovered.

In Version 1 of the SAT all the class bits were first saved, then during stage two summing all the class bits were examined to determine the weights to sum. This was the bottleneck discussed in the introduction. In general, L is chosen to be $\log_2 N$ of the class size [11], which means that relatively few bits are set and most of the class bits examined will be zero. To take advantage of this, Version 2 of the SAT stores the relative address of the class bit instead of the class bits themselves, as proposed in [7]. This is shown in Figure 6 where the values 0, 3, 9 etc. are stored in the buffer memory instead of the entire class pattern.

Fig. 6: Class Bit Relative Address Storage



Stage Two Summing and Thresholding

The basic hardware for stage two is the same as that used for the stage one summing. The only significant difference between the two operations is that the summed values are not written to the buffer memory unless the user specifically requests it.

4.1. SAT Performance Evaluation

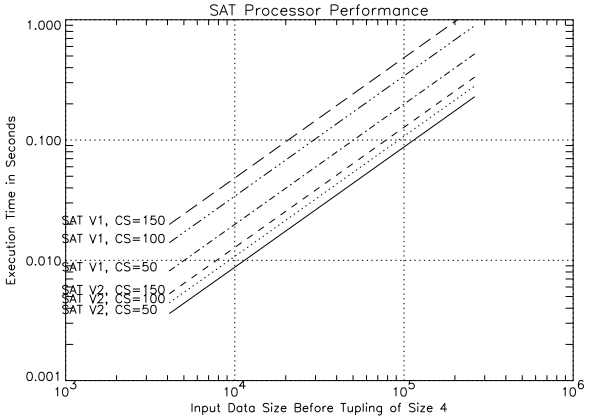
The timing evaluation has been performed by analysing the state machine behaviour to derive Equation 1 which can be used to determine the execution time for any input data using the following variables:

- α - Class size
- β - Stage 1 input image size
- δ - Stage 1 tuple size
- ι - Number of iterations required to find all L bits of the class pattern
- ϕ - How often a summed value equals the stored threshold value per iteration
- ρ - Stage 2 input image size
- σ - Stage 2 tuple size
- τ - Number of bits set in the class pattern (usually τ = L)

4.2. SAT Processor Comparison

Using Equation 1 the new SAT was compared with Version 1 to produce the graph of Figure 7. To produce the graph a tuple size of four was used (a typical size used in many applications [9]), the number of iterations was one. The graph shows the speed of operation of SAT Versions 1 and 2, for a range of input sizes and for class sizes of 50, 100 and 150 bit class sizes, with the number of class bits set to $\log_2 CLASS\_SIZE$. In order to show the design benefits of the new SAT processor instead of just the improvements in technology used (ie. faster memory) the old design speeds have been shown scaled as if it was also using the new 25ns memory.

Fig. 7: SAT Execution Time Analysis



From the results used to generate Figure 7 it is calculated that Version 2 of the SAT ranges between 2.25 times faster than Version 1 for a class size of 50 and 3.8 times faster than Version 1 for a class size of 150. Kennedy [7] showed that Version 1 of the SAT was 48 times faster than the standard DSP and its dedicated coprocessor. This suggests that Version 2 of the SAT processor is now between 108 times and 182 times faster than the DSP with its dedicated coprocessor.

It is interesting to consider how large an image could be processed using 25Hz frames per second (fps) input images on a *single* C node. It was shown in [7] that the maximum size input image that the DSP with the old dedicated coprocessor could process is 22 x 22 pixels at 25 fps using a tuple size of four and a class size of 32 bits; note that the DSP would also have to perform pre-processing operations, such as the tupling, that would further reduce the class size or the image size. The new version of the SAT processor can theoretically process an input image of 230 x 230 pixels, while freeing the DSP to perform the pre-processing and memory

$$Execution\ Time(seconds) = 50nanoseconds \times \left[\frac{\alpha}{16}\left(\frac{3.5\beta}{\delta}+34\right)+\left(\left(\frac{\rho}{\sigma}\times 2^{\sigma}\right)\times(3.5\tau+35)\right)+\iota(4.5\alpha+3\phi)\right] \quad (1)$$

transfer operations. This is considered a significant improvement.

Benchmarking the SAT Processor

To appreciate the benefits of the SAT processor it has been compared with a typical fast workstation, a Silicon Graphics R4600SC Indy workstation that has a 512K secondary cache and a processor clock speed of 133 MHz. The workstation takes 1790ns to sum 32 bits of matrix data whilst the SAT takes 350ns, making the SAT faster than the workstation by a factor of 5 times. This is a large degree of speed-up considering that the SG workstation costs around £7000 per unit whilst the SAT daughter board costs in the region of £300.

The SAT would benefit from implementation in VLSI as this would allow the data width to be increased to 32 or 64 bits providing an immediate speed-up over the workstation of 10 or 20 times respectively. VLSI would also allow the SAT clock speed to be greatly increased.

5. Conclusion

This paper has described Version 2 of the C-NNAP parallel processor and Version 2 of the SAT processor. It has been seen that the SAT operates between two and four times faster than Version 1 of the SAT processor. The analysis has shown that the DSP with coprocessor can process a 22 x 22 image whilst SAT Version 2 can process a 230 x 230 pixel image, considering 25 frames per second image processing. These improvements will allow the users of the C-NNAP machine to develop more sophisticated real-time image processing applications than was previously possible.

References

[1] A. W. Andersen, S. S. Christensen, and T. M. Jorgensen, "An active vision system for robot guidance using a low cost neural network board," in *European Robotics and Intelligent Systems Conference*, (Malaga, Spain), pp. 480–488, August 1994.

[2] J. Austin, "The cellular neural network associative processor, C-NNAP," in *Fifth International Conference on Image Processing and its Applications*, pp. 622–626, July 1995.

[3] J. Austin, "Distributed associative memories for high speed symbolic reasoning," *International Journal on Fuzzy Sets and Systems*, 1995. Invited paper to the special issue on Connectionist and Hybrid Connectionist Systems for Approximate Reasoning.

[4] J. Austin and S. Buckle, "Segmentation and matching in infra-red airborne images using a binary neural network," in *Neural Networks*, (J. Taylor, ed.), ch. 8, pp. 95–118, Alfred Waller, 1995.

[5] J. Austin and T. J. Stonham, "An associative memory for use in image recognition and occlusion analysis," in *Image and Vision Computing*, pp. 251–261, November 1987.

[6] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Proceedings of the Joint Computer Conference*, 1959.

[7] J. V. Kennedy, *A Hardware Implementation of a Dedicated Associative Processor*. Master's thesis, University of York, UK, March 1994.

[8] J. V. Kennedy and J. Austin, "A hardware implementation of a binary neural associative memory," in *Fourth International Conference on Microelectronics for Neural Networks And Fuzzy Systems*, (Torino, Italy), pp. 178–185, September 1994.

[9] S. E. M. O'Keefe and J. Austin, "Application of an associative memory to the analysis of document fax images," in *The British Machine Vision Conference*, (York, UK), pp. 315–326, 1994.

[10] R. Pack, "DSPVME, a digital signal processor based VME bus system for neural/image processing," Tech. Rep., University of York, UK, 1990.

[11] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, pp. 960–962, June 1969.